Using PageRank Algorithm to Improve Coupling Metrics

Cheolhyun Park, Junhee Kim, and Eunseok Lee
Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, Republic of Korea
Email: {pch851130, comemail, leees}@skku.edu

***Abstract*—Existing coupling metrics only use the number of methods invocations, and does not consider the weight of the methods. Thus, they cannot measure coupling metrics accurately. In this paper, we measure the weight of methods using PageRank algorithm, and propose a new approach to improve coupling metrics using the weight. We validate the proposed approach by applying them to several open source projects. And we measure several coupling metrics using existing approach and proposed approach. As a result, the correlation between change-proneness and improved coupling metrics were significantly higher than existing coupling metrics. Hence, our improved coupling metrics can more accurately measure software.**

***Index Terms*—software quality metrics, coupling metrics, PageRank algorithm**

I. Introduction

Software quality measurement is essential in software engineering. Coupling of software quality metrics is "the measure of the strength of association established by a connection from one module to another" [1]. Coupling is used for various purposes, such as modularization, reusability, fault-proneness, and change-proneness. In this paper, we propose a new approach to improve coupling metrics. Existing coupling metrics mainly use the number of method invocations [2]. These coupling metrics does not consider the weight of methods; assume that all methods have same weight. But since there are any methods that are more important than other methods, considering the weight of methods is more accurate. In this paper, we measure the weight of methods using PageRank algorithm, and propose a new approach to improve coupling metrics using the weight. We validate the proposed approach by applying them to several open source projects. And we measure several coupling metrics using existing approach and proposed approach. As a result, the correlation between change-proneness and improved coupling metrics were significantly higher than existing coupling metrics. We observed that improved coupling metrics could more accurately measure software quality than existing coupling metrics. The rest of the paper is organized as follows. Section 2 describes the existing coupling metrics; In Section 3, we outline our new approach to measure the weight of methods using PageRank algorithm for improve coupling metrics. Section 4 presents a case study as an empirical evaluation of the proposed approach. Section 5 concludes the paper and outlines future research.

II. Related Work and Background

A. Structural Coupling Metrics

Existing coupling metrics only consider the structural relationship of software. CBO and COF measures coupling using call relationship between classes [3, 4, 5]. RFC measures coupling using the number of methods of a class and called methods of a class [3, 4]. MPC measures coupling using the number of called methods of a class [6]. DAC measures coupling using the number of class variables defined in a class [6]. In this paper, we improve four coupling metrics (CBO, MPC, RFC, and COF). Since these coupling metrics is directly consider the number of methods invocations, we select these metrics.

B. Semantic Coupling Metrics

Structural metrics only measures coupling from a data-flow perspective. Semantic metrics can measure coupling from a semantic perspective, such as functional. CoCC (Conceptual Coupling of a Class) measures coupling using conceptual similarity between elements of source code [7]. RTC (Relational Topic based Coupling) measure coupling using Relational Topic Models, generative probabilistic to capture latent topics in source code [8]. Since semantic metrics is only applicable to source code, these are not applicable to artifacts (i.e., method call graph) in design phase. In this paper, we introduce a new approach to improve coupling metrics that are applicable to artifacts in design phase.

C. PageRank Algorithm

PageRank is link analysis algorithm that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web [9]. In this paper, we measure weight of methods using PageRank algorithm. Equation (1) shows that calculation method of PageRank algorithm.

$$PR(A) = (1-d) + d(PR(T_1)/C(T_1)+...+PR(T_n)/C(T_n)) \quad (1)$$

$d = 0.85$

$C(T_i)$: the number of outbound links on page T_i

Figure 1 is webpage link graph that is generated randomly. This graph contains the nodes (webpages) and edges (link relationship between webpages). The initial value of PageRank for each page is assigned to 0.

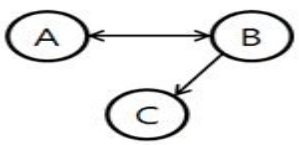


Figure 1. Webpage link graph

DOI: 01.IJIT.02.01.56

Since the calculation is an iterative process, the algorithm is stopped after a given count of iterations or if the PageRank value differences between iterations are less than predefined values.

TABLE I. RESULTS OF PAGERANK ALGORITHM

iterate	PR(A)	PR(B)	PR(C)
1	0.15	0.28	0.27
2	0.27	0.38	0.31
3	0.31	0.41	0.33
4	0.33	0.43	0.33
5	0.33	0.43	0.33

III. PROPOSED APPROACH

In this section, we measure the weight of methods using PageRank algorithm, and improve four coupling metrics using the measured weight of methods.

A. Measurement of Methods Weight

In order to measure coupling metrics, figure 2 is method call graph generated randomly. This graph consists of nodes(methods) and edges(call relationship between methods).

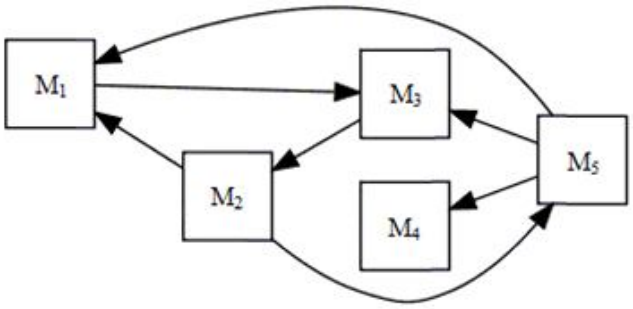


Figure 2. Method call graph

We applied Pagerank algorithm to this method call graph. Table 2 shows that weight of five methods were computed based on PageRank algorithm.

TABLE II. WEIGHT OF METHODS

PR(M_1)	PR(M_2)	PR(M_3)	PR(M_4)	PR(M_5)
0.68	0.90	0.88	0.30	0.53

Table 3 shows methods of each class. We can construct class call graph using Figure2 and Table 3. Figure 3 shows class call graph.

TABLE III. METHODS OF EACH CLASS

	methods
C_1	M_1, M_2
C_2	M_3, M_4
C_3	M_5

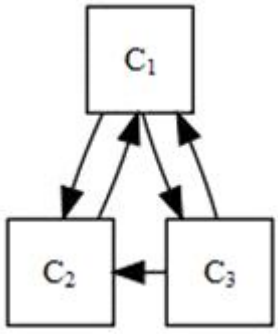


Figure 3. Class call graph

Table 4 shows that weight of three classes were computed based on PageRank algorithm. As mentioned above, we can measure weight of methods and classes using PageRank algorithm.

TABLE IV. WEIGHT OF CLASSES

PR(C_1)	PR(C_2)	PR(C_3)
1.30	1.00	0.70

A. Improvement of Coupling Metrics

In this section, coupling metrics can be described by following equations. When we use suffix 'N' in each metrics, it means improved cohesion metrics.

1) CBO

$$CBO(c) = |SIC(c)| \quad (2)$$

$$CBO_N(c) = \sum_{c' \in SIC(c)} PR(c') \quad (3)$$

$SIC(c)$: The set of called classes of class 'c'

$PR(c)$: PageRank of class 'c'

2) MPC

$$MPC(c) = |SIM(c)| \quad (4)$$

$$MPC_N(c) = \sum_{m' \in SIM(c)} PR(m') \quad (5)$$

$SIM(c)$: The set of called methods of class 'c'

$PR(m)$: PageRank of method 'm'

3) RFC

$$RFC(c) = |SIM(c)| + M \quad (6)$$

$$RFC_N(c) = \sum_{m' \in SIM(c)} PR(m') + \sum_{i=1}^{M} PR(m_i) \quad (7)$$

M : The number of methods of class 'c'

4) CFO

$$COF(c) = \frac{|SIC(c)|}{C^2 - C} \quad (8)$$

$$COF_N(c) = \frac{\sum_{c' \in SIC(c)} PR(c')}{C^2 - C} \quad (9)$$

C : The number of all classes

DOI: 01.IJIT.02.01.56

TABLE V. OPEN SOURCE PROJECTS

Project	Used version	Description	Classes	LOC	Changed LOC
Robocode	1.3→1.7	Programming game	166	39991	3659
JHotDraw	7.1→7.5	Java GUI framework	150	29667	3408
JabRef	1.0→2.7	Bibliography reference manager	75	29182	1232
Heritrix	1.0→1.4	Web crawler	196	58288	3712

IV. CASE STUDY

We validate the proposed approach by applying them to four open source projects. Table 5 gives a description of four open source project.

A. Experimental Process

1) Java parser generates class information. And we can measure coupling metrics of each class using created class information.

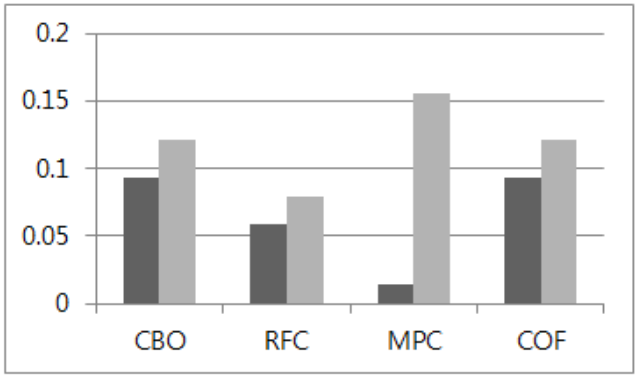
2) Then we count the number of changed lines in each class. And we computed change-proneness using the number of changed lines. In this paper, we define change-proneness as equation (10) that is independent of source code size.

$$\text{Change proneness} = \frac{number\ of\ changed\ lines}{number\ of\ all\ lines} \quad (10)$$

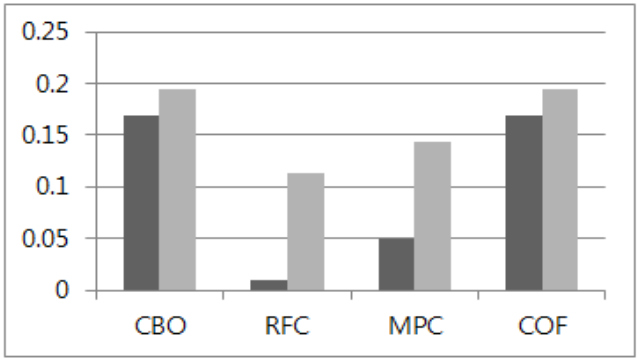
3) Finally, we computed Pearson's correlation coefficient between change-proneness and coupling metrics, for each class. Generally, coupling metrics is used as predictor of the change-proneness of a class[10, 11, 12]. Therefore, in order to empirically confirm superiority of improved coupling metrics, we computed correlation coefficient between change-proneness and coupling metrics.

B. Experimental Results

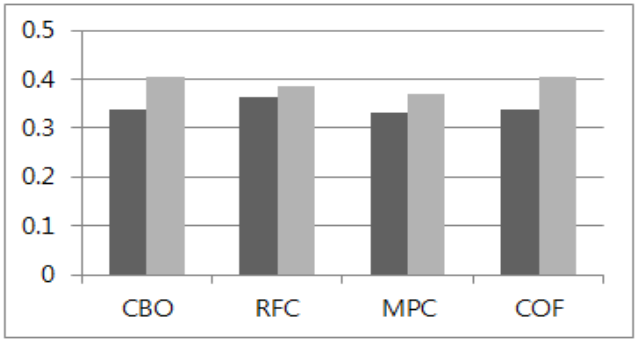
We measured four coupling metrics of each class with existing and proposed approach. And we compute the correlation coefficient between change-proneness and each coupling metrics. We show the results of this experiment in figure 4. In this figure 4, the first bar represents the correlation coefficient between change-proneness and existing coupling metrics, and the second bar represents the correlation coefficient between change-proneness and improved coupling metrics. In four open source projects, we observed that the correlation coefficient of all coupling metrics has been improved. Our improved coupling metrics showed significantly an improved correlation coefficient as compared to the existing coupling metrics. Thus, improved coupling metrics proved to be better predictors than existing coupling metrics. It can be concluded that improved coupling metrics is the better metrics.



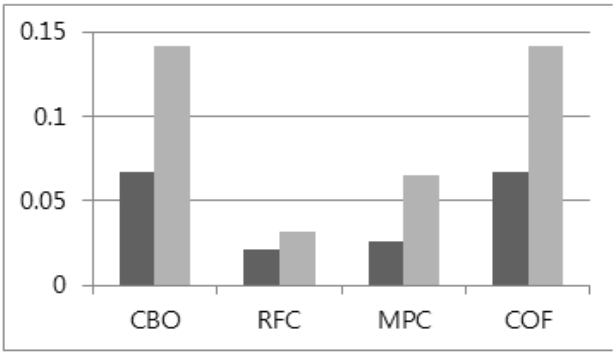
(a) Robocode 1.3 →1.7



(b) JHotDraw 7.1 →7.5



(c) JabRef 1.0 → 2.7



(d) Heritrix 1.0 →1.4

DOI: 01.IJIT.02.01.56

V. Conclusion And Future Work

In this paper, we measure the weight of methods using PageRank algorithm, and proposed a new approach to improve coupling metrics using the weight. We applied improved coupling metrics to four open source projects, measured the correlation coefficient between change-proneness and coupling metrics, and empirically observed superiority of proposed approach. In the future work, we will explore new approach to measure the weight of source code lines, and improve complexity metrics using the weight.

References

[1] W. Stevens, G. Myers, and L. Constantine, "Structured Design," IBM Systems J., vol. 13, no. 2, pp. 115-139, 1974.

[2] L. C. Briand, J. W. Daly, and J. Wüst, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," IEEE Transactions on Software Engineering, vol. 25, pp. 91-121, 1999.

[3] S.R. Chidamber and C.F. Kemerer, "Towards a Metrics Suite for Object Oriented Design," A. Paepcke, ed., Proc. Conf. Object-Oriented Programming: Systems, Languages and Applications, OOPSLA' 91, Oct. 1991. Also published in SIGPLAN Notices, vol. 26, no. 11, pp. 197-211, 1991.

[4] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476-493, 1994.

[5] F. Abreu, M. Goulão, and R. Esteves, "Toward the Design Quality Evaluation of Object-Oriented Software Systems," Proc. Fifth Int'l Conf. Software Quality, Austin, Texas, Oct. 1995.

[6] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," J. Systems and Software, vol. 23, no. 2, pp. 111-122, 1993.

[7] D. Poshyvanyk and A. Marcus, "The Conceptual Coupling Metrics for Object Oriented Systems," Proc. 22nd IEEE International Conference on Software Maintenance (ICSM'06), 2006, pp. 469-478.

[8] M. Gethers and D. Poshyvanyk. "Using relational topic models to capture coupling among classes in object-oriented software systems." In ICSM'10, 2010.

[9] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Libraries Working Paper, 1998.

[10] F. G. Wilkie, B. A. Kitchenham, "Coupling Measures and Change Ripples in C++ Application Software", published in the proceedings of EASE'99, University of Keele, UK, 1998.

[11] Kagdi H, Gethers M, Poshyvanyk D, Collard M (2010) "Blending conceptual and evolutionary couplings to support change impact analysis in source code", 17th IEEE Working Conference on Reverse Engineering (WCRE'10). Boston, USA, 119–128.

[12] R. Robbes, D. Pollet, and M. Lanza. "Logical coupling based on fine-grained change information", In Proceedings of WCRE 2008 (15th Working Conference on Reverse Engineering), 2008.

DOI: 01.IJIT.02.01.56